Grafika Komputerowa i Komunikacja Człowiek-Komputer

Data zajęć: 3.12.2018 Data oddania: 23.12.2018
Termin zajęć: Pon. TP 9:15 Prowadzący zajęcia: Mgr inż. Szymon Datko

Sprawozdanie nr 3

Jakub Majewski 238902

---

1. Opis tematu

Temat: OpenGL - podstawowa obsługa kamery i prosty system zdarzeń (obsługa myszy).

Zrealizowane zadania:
1. Zapoznanie się z podstawowymi funkcjami OpenGL odpowiadającymi za:
   a. obsługę kamery.
   b. obsługę myszy.
2. Napisanie programu umożliwiającego obracanie i "zoomowanie" dowolnego obiektu 3D za pomocą myszy. 2. Napisanie programu umożliwiającego obracanie i "zoomowanie" kamery obserwującej dowolny obiekt 3D.

2. Opis najważniejszych fragmentów kodu

1. Metoda wykonująca się przy dowolnym zdarzeniu związanym ze zmianą stanu dowolnego przycisku myszy, tzw. "callback" czyli odpowiedź w postaci metody wywoływanej na konkretne zdarzenie (ang. event):

```
1  void Mouse(int btn, int state, int x, int y)
2  {
3    if (state == GLUT_DOWN) {
4
5      x_pos_old = x; // zapamiętanie aktualnej pozycji x kursora
6      y_pos_old = y; // zapamiętanie aktualnej pozycji y kursora
7
8          // zmienna 'status' odpowiada za przechowywanie ID
           ↪  wciśniętego klawisza myszy
9          // status == 0 oznacza, że nie został wciśnięty żaden
           ↪  klawisz
10         // status != 0 oznacza, że jakiś klawisz jest trzymany
11         // ważne: nie rozpoznaje statusu kliknięcia i
           ↪  "odkliknięcia"
12     if (btn == GLUT_LEFT_BUTTON) status = 1;
13     else if (btn == GLUT_RIGHT_BUTTON) status = 2;
14   }
15   else status = 0; // nie został wciśnięty żaden klawisz
16 }
```

2. Callback dla dowolnego zdarzenia związanego ze zmianą położenia kursora (np. ruch myszą).

```
1  void Motion(GLsizei x, GLsizei y)
2  {
3
4    delta_x = x - x_pos_old;  // obliczenie różnicy położenia
     ↪  kursora myszy
5    x_pos_old = x;            // podstawienie bieżącego położenia
     ↪  jako poprzednie
6
7    delta_y = y - y_pos_old;  // obliczenie różnicy położenia
     ↪  kursora myszy
8    y_pos_old = y;            // podstawienie bieżącego położenia
     ↪  jako poprzednie
9
10   glutPostRedisplay();      // przerysowanie obrazu sceny
11 }
```

3. Metoda aktualizująca pozycję kamery (zmienia tylko wartości) tak aby znajdowała się ona na sferze o promieniu 'zoom' i pozycji środka równej pozycji obiektu obserwowanego (tutaj 0,0,0)

```
void cameraPositionUpdate() {
  viewer[0] = zoom * std::cosf(angleX) * std::cosf(angleY);
  viewer[1] = zoom * std::sinf(angleY);
  viewer[2] = zoom * std::sinf(angleX) * std::cosf(angleY);
}
```

4. Ustawianie pozycji kamery OpenGLa na viewer[0..2] oraz ustawianie orientacji dla osi pionowej (jest to łatka naprawiająca błąd przeskakującej i wariującej kamery po wykonaniu obrotu o 180 stopni wokół osi OX).

```
void RenderScene(void) {
    //...
  gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0, 0,
  ↪  top, 0);
    //...
}
```

5. Obliczanie wartości zmiennej 'top' (wykorzystanej w poprzedniej metodzie) oraz ustawianie ograniczeń i zmniejszenie "czułości" myszy podczas ruchu kamery. void eventUpdate()

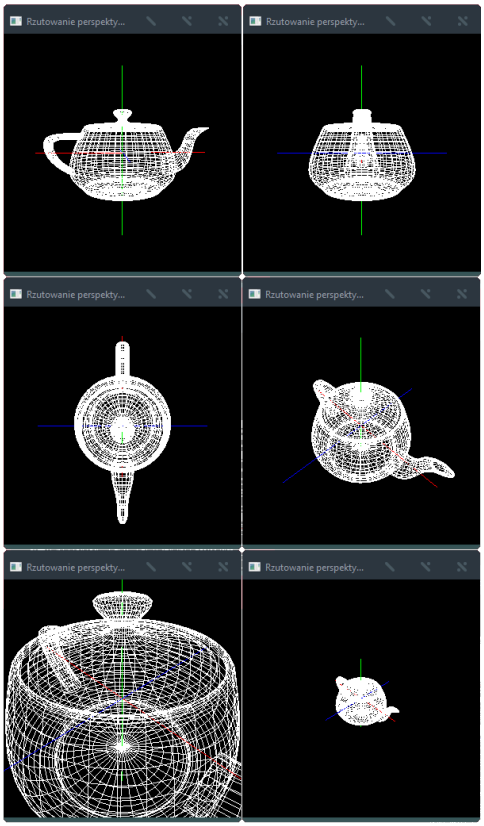
float dpi = 0.05;

if (status == 1) angleX += delta$_x * pix2angle * dpi; angleY += delta_y * pix2angle * dpi$;

top = 1.0 - 2.0 * (std::fmod(angleY + 3.14159 * 0.5, 3.14159*2.0) > 3.14159);

// Powyższy zapis w pseudokodzie mógłby wyglądać tak: // top = 1.0; // // Sprawdzanie czy kamera obserwuje "tył" obiektu // // innymi słowy czy obróciła się o ponad 90 stopni w górę albo w dół) // if ((angleY + 90 stopni) mod (360 stopni)) > 180 stopni: / // top = -1.0; else if (status == 2) zoom += delta$_y * pix2angle * dpi; if(zoom < 0.05) zoom = 0.05; elseif(zoom > 20.0) zoom = 20.0; cameraPositionUpdate()$;

3. Rezultat pracy



Rysunek 1: Obiekt obserwowany pod różnymi kątami kamery

4. Spostrzeżenia i wnioski

Największym problemem podczas pisania kodu okazało się naprawienie błędu (obliczenie wartości zmiennej 'top') o którym mowa w punkcie 4. i 5. w sekcji "najważniejsze fragmenty kodu".